```
# -*- coding:utf-8 -*-
import tkinter
import tkinter.messagebox
import random

# キャンバスの横方向・縦方向のサイズ（px）
CANVAS_SIZE = 400

# ラインの数
NUM_LINE = 10

# 色の設定
BOARD_COLOR =”burlywood3” # 盤面の背景色
YOUR_COLOR = ”black” # あなたの石の色
COM_COLOR = ”white” # 相手の石の色

# プレイヤーを示す値
YOU = 1
COM = 2

class Gobang():
    def __init__(self, master):
        '''コンストラクタ'''
        self.master = master # 親ウィジェット
        self.player = YOU # 次に置く石の色
        self.board = None # 盤面上の石を管理する 2 次元リスト
        self.color = { # 石の色を保持するリスト
            YOU : YOUR_COLOR,
            COM : COM_COLOR
        }
        self.nextDisk = None

        # ウィジェットの作成
        self.createWidgets()

        # イベントの設定
        self.setEvents()

        # 五目並べゲームの初期化
        self.initGobang()


```

```
43  def createWidgets(self):
44      '''ウィジェットを作成・配置する'''
45      # キャンバスの作成
46      self.canvas = tkinter.Canvas(
47          self.master,
48          bg=BOARD_COLOR,
49          width=CANVAS_SIZE,
50          height=CANVAS_SIZE,
51          highlightthickness=0
52      )
53      self.canvas.pack(padx=10, pady=10)
54
55  def setEvents(self):
56      '''イベントを設定する'''
57      # キャンバス上のマウスクリックイベントを受け付ける
58      self.canvas.bind('<ButtonPress>', self.click)
59
60
61  def initGobang(self):
62      '''ゲームの初期化を行う'''
63      # 盤面上の石を管理する 2 次元リストを作成（最初は全てNone）
64      self.board = [[None] * (NUM_LINE) for i in range(NUM_LINE)]
65
66      # 線と線の間隔（px）を計算
67      self.interval = CANVAS_SIZE // (NUM_LINE + 1)
68
69      # 交点描画位置の左上オフセット
70      self.offset_x = self.interval
71      self.offset_y = self.interval
72
73      # 縦線を描画
74      for x in range(NUM_LINE):
75          # 線の開始・終了座標を計算
76          xs = x * self.interval + self.offset_x
77          ys = self.offset_y
78          xe = xs
79          ye = (NUM_LINE - 1) * self.interval + self.offset_y
80
81          # 線を描画
82          self.canvas.create_line(
83              xs, ys,
84              xe, ye,
```

```
85          )
86
87      # 横線を描画
88      for y in range(NUM_LINE):
89          # 線の開始・終了座標を計算
90          xs = self.offset_x
91          ys = y * self.interval + self.offset_y
92          xe = (NUM_LINE - 1) * self.interval + self.offset_x
93          ye = ys
94
95          # 線を描画
96          self.canvas.create_line(
97              xs, ys,
98              xe, ye,
99          )
100
101     def drawDisk(self, x, y, color):
102         '''(x,y)の交点に色がcolorの石を置く（円を描画する）'''
103         # (x,y)の交点の中心座標を計算
104         center_x = x * self.interval + self.offset_x
105         center_y = y * self.interval + self.offset_y
106
107         # 中心座標から円の開始座標と終了座標を計算
108         xs = center_x - (self.interval * 0.8) // 2
109         ys = center_y - (self.interval * 0.8) // 2
110         xe = center_x + (self.interval * 0.8) // 2
111         ye = center_y + (self.interval * 0.8) // 2
112
113         # 円を描画する
114         tag_name = 'disk_' + str(x) + '_' + str(y)
115         self.canvas.create_oval(
116             xs, ys,
117             xe, ye,
118             fill=color,
119         )
120
121         return tag_name
122
123     def getIntersection(self, x, y):
124         '''キャンバス上の座標を交点の位置に変換'''
125         ix = (x - self.offset_x + self.interval // 2) // self.interval
126         iy = (y - self.offset_y + self.interval // 2) // self.interval
```

```
127
128         return ix, iy
129
130     def click(self, event):
131         '''盤面がクリックされた時の処理'''
132         if self.player != YOU:
133             # COMが石を置くターンの時は何もしない
134             return
135
136         # クリックされた位置がどの交点であるかを計算
137         x, y = self.getIntersection(event.x, event.y)
138         if x < 0 or x >= NUM_LINE or y < 0 or y >= NUM_LINE:
139             # 盤面外の交点の場合は何もしない
140             return
141
142         if not self.board[y][x]:
143             # 石が置かれていない場合はクリックされた位置に石を置く
144
145             # 石を置く
146             self.place(x, y, self.color[self.player])
147
148     def place(self, x, y, color):
149         '''(x,y)の交点に色がcolorの石を置く'''
150         # (x,y)に石を置く（円を描画する）
151         self.drawDisk(x, y, color)
152
153         # 描画した円の色を管理リストに記憶させておく
154         self.board[y][x] = color
155
156         # 5つ並んだかどうかをチェック
157         if self.count(x, y, color) >= 5:
158             self.showResult()
159             return
160
161         # プレイヤーは交互に変更
162         if self.player == COM:
163             self.player = YOU
164         else:
165             self.player = COM
166
167         if self.player == COM:    # 次のプレイヤーがCOMなら1秒後に石を置く
168             self.master.after(1000, self.com)
```

```
169
170
171     def count(self, x, y, color):
172         '''(x,y)に色がcolorの石を置いた時の石の並び数をチェック'''
173         # チェックする方向をリストに格納
174         count_dir = [
175             (1, 0), # 右
176             (1, 1), # 右下
177             (0, 1), # 上
178             (-1, 1), # 左下
179         ]
180
181         max = 0 # 石の並び数の最大値
182
183         # count_dirの方向に対して石の並び数をチェック
184         for i, j in count_dir:
185
186             # 石の並び数を1に初期化
187             count_num = 1
188
189             # (x,y)から現在の方向に対して 1 交点ずつ石が連続しているかをチェック
190             for s in range(1, NUM_LINE):
191                 xi = x + i * s
192                 yj = y + j * s
193                 if xi < 0 or xi >= NUM_LINE or yj < 0 or yj >= NUM_LINE:
194                     # 盤面外の交点の場合は石は連続していない
195                     break
196
197                 if self.board[yj][xi] != color:
198                     # 異なる色の石が置かれていれば石は連続していない
199                     break
200
201                 # 上記以外の場合は石が連続している
202                 count_num += 1
203
204             # 次は逆方向をチェック
205             for s in range(-1, -(NUM_LINE), -1):
206                 xi = x + i * s
207                 yj = y + j * s
208                 if xi < 0 or xi >= NUM_LINE or yj < 0 or yj >= NUM_LINE:
209                     break
210
```

```
211             if self.board[yj][xi] != color:
212                 break
213 
214             count_num += 1
215 
216         # 最大値の置き換え
217         if max < count_num:
218             max = count_num
219 
220     # 石が連続している数の最大値を返却
221     return max
222 
223 def showResult(self):
224     '''ゲーム終了時の結果を表示する'''
225     # 勝利者は先ほど石を置いたプレイヤー
226     winner = self.player
227 
228     # 結果をメッセージボックスで表示する
229     if winner == YOU:
230         tkinter.messagebox.showinfo("結果" ,"あなたの勝ちです!!!")
231     else:
232         tkinter.messagebox.showinfo("結果","あなたの負けです…")
233 
234 def com(self):
235     '''COMに石を置かせる'''
236     # 相手が石を置いた時に石が最大で連続する交点の座標を取得
237     max_list = []
238     max = 0
239     for y in range(NUM_LINE):
240         for x in range(NUM_LINE):
241             if not self.board[y][x]:
242                 # (x,y)座標に相手が石を置いた場合に石が連続する数を取得
243                 count_num = self.count(x, y, self.color[YOU])
244                 if count_num == max:
245                     max_list.append((x, y))
246                 elif count_num > max:
247                     max_list = []
248                     max_list.append((x, y))
249                     max = count_num
250 
251     # 石が連続する数が最大になる交点の中から１つの座標をランダムに取得
252     choice = random.randrange(len(max_list))
```

```
253         x, y = max_list[choice]
254 
255         # 石を置く
256         self.place(x, y, COM_COLOR)
257 
258 # スクリプト処理ここから
259 app = tkinter.Tk()
260 app.title('五目並べ')
261 gobang = Gobang(app)
262 app.mainloop()
263 
264 
```